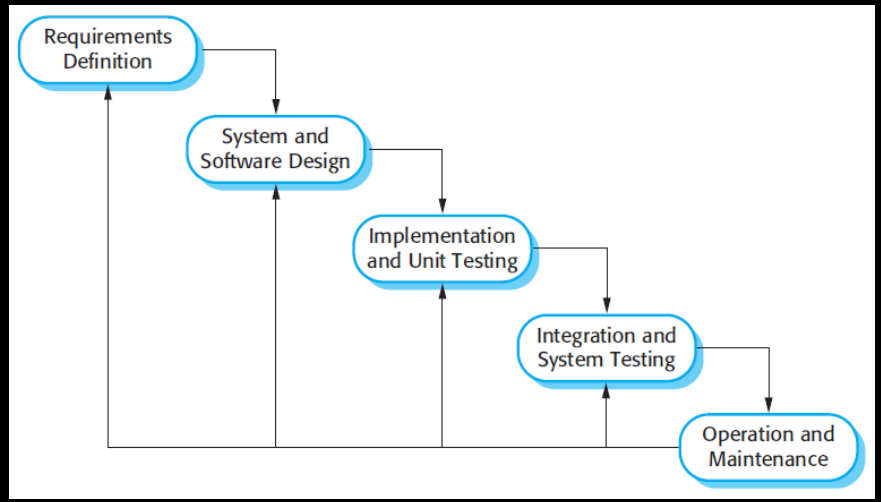
UNIT - 2 : PROCESS MODELS



CONTENTS :

- The waterfall model
- Incremental process models
- Evolutionary process models
- Specialized Process Models
- Agile process - Agile process Model: Extreme programming

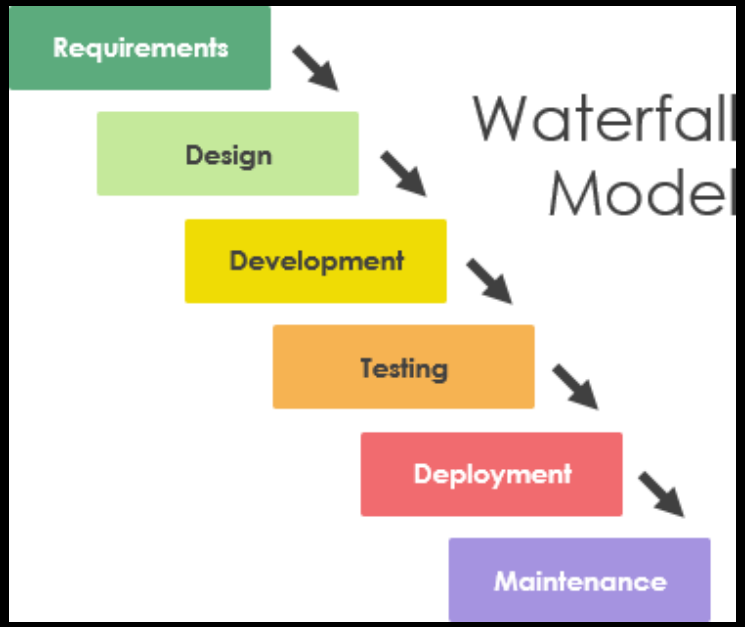
PROCESS MODELS

- In software engineering, **process models** serve as frameworks that define the sequence of activities, tasks, and deliverables required to develop high-quality software. These models act as roadmaps, providing step-by-step guidance for software development teams.
- Software processes, methodologies and frameworks range from specific prescriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a "sponsor" or "maintenance" organization distributes an official set of documents that describe the process.

- **Software Process and Software Development Lifecycle Model :**
- One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. The most used, popular and important SDLC models are given below:
- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

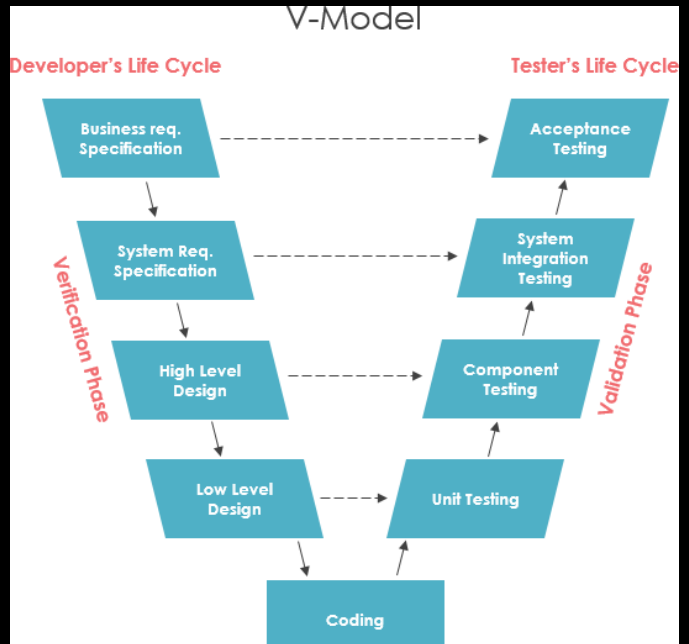
WATERFALL MODEL:

- The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialisation of tasks. The approach is typical for certain areas of engineering design.
-



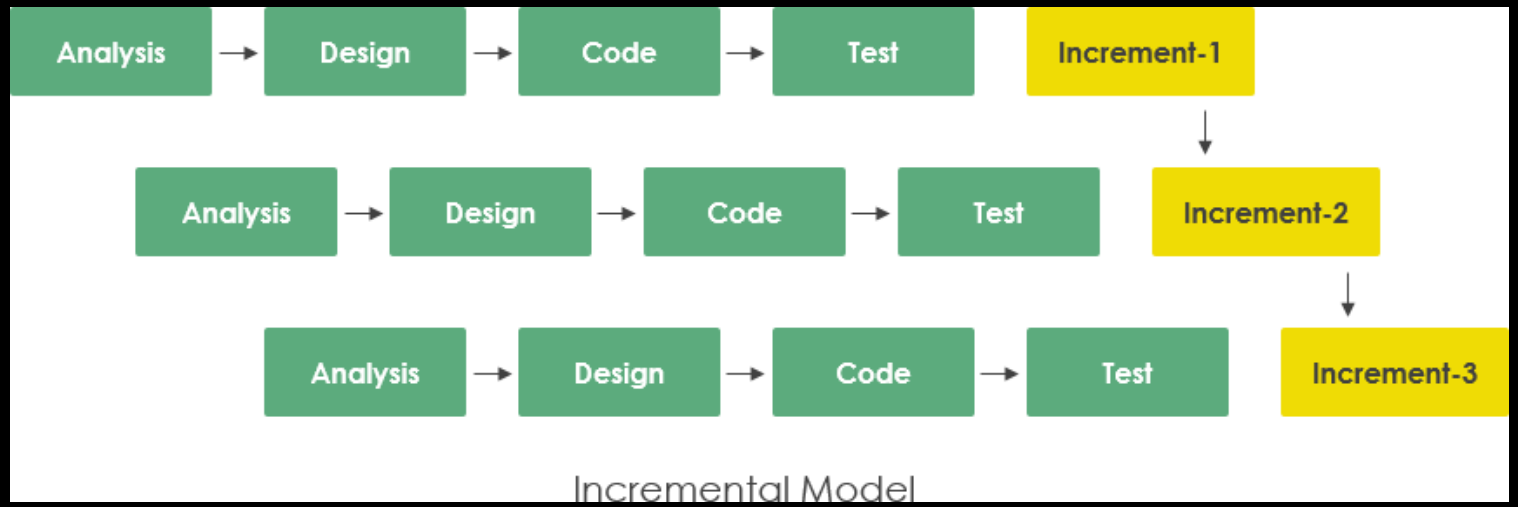
V MODEL :

- The V-model represents a development process that may be considered an extension of the waterfall model and is an example of the more general V-model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represent time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.



INCREMENTAL MODEL :

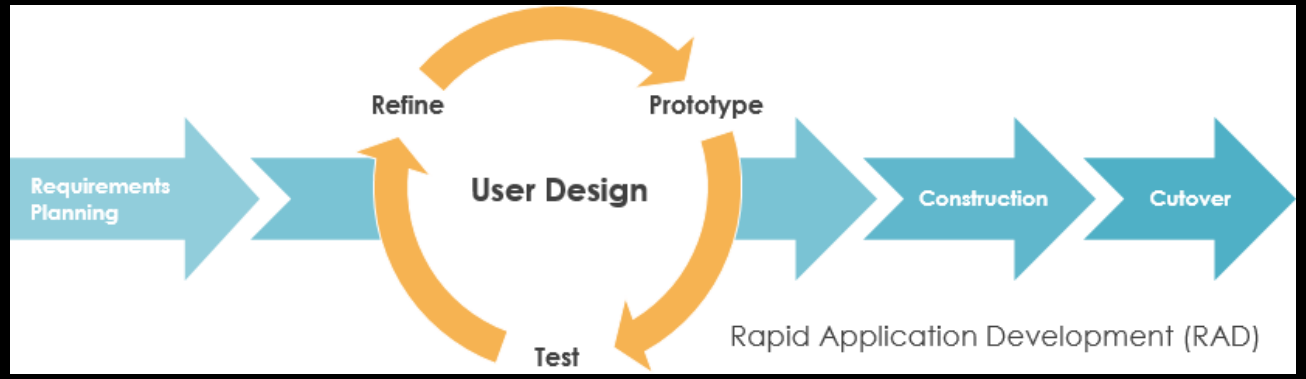
- The incremental build model is a method of software development where the model is designed, implemented and tested incrementally (a little more is added each time) un the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionally has beer implemented. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.



Incremental Model

RAD MODEL

- Rapid application development was a response to plan-driven waterfall processes, developed in the 1970s and 1980s, such as the Structured Systems Analysis and Design Method (SSADM). Rapid application development (RAD) is often referred as the adaptive software development. RAD is an incremental prototyping approach to software development that end users can produce better feedback when examining a live system, as opposed to working strictly with documentation. It puts less emphasis on planning and more emphasis on an adaptive process.
- RAD may resulted in a lower level of rejection when the application is placed into production, but this success most often comes at the expense of a dramatic overruns in project costs and schedule. RAD approach is especially well suited for developing software that is driven by user interface requirements. Thus, some GUI builders are often called rapid application development tools.



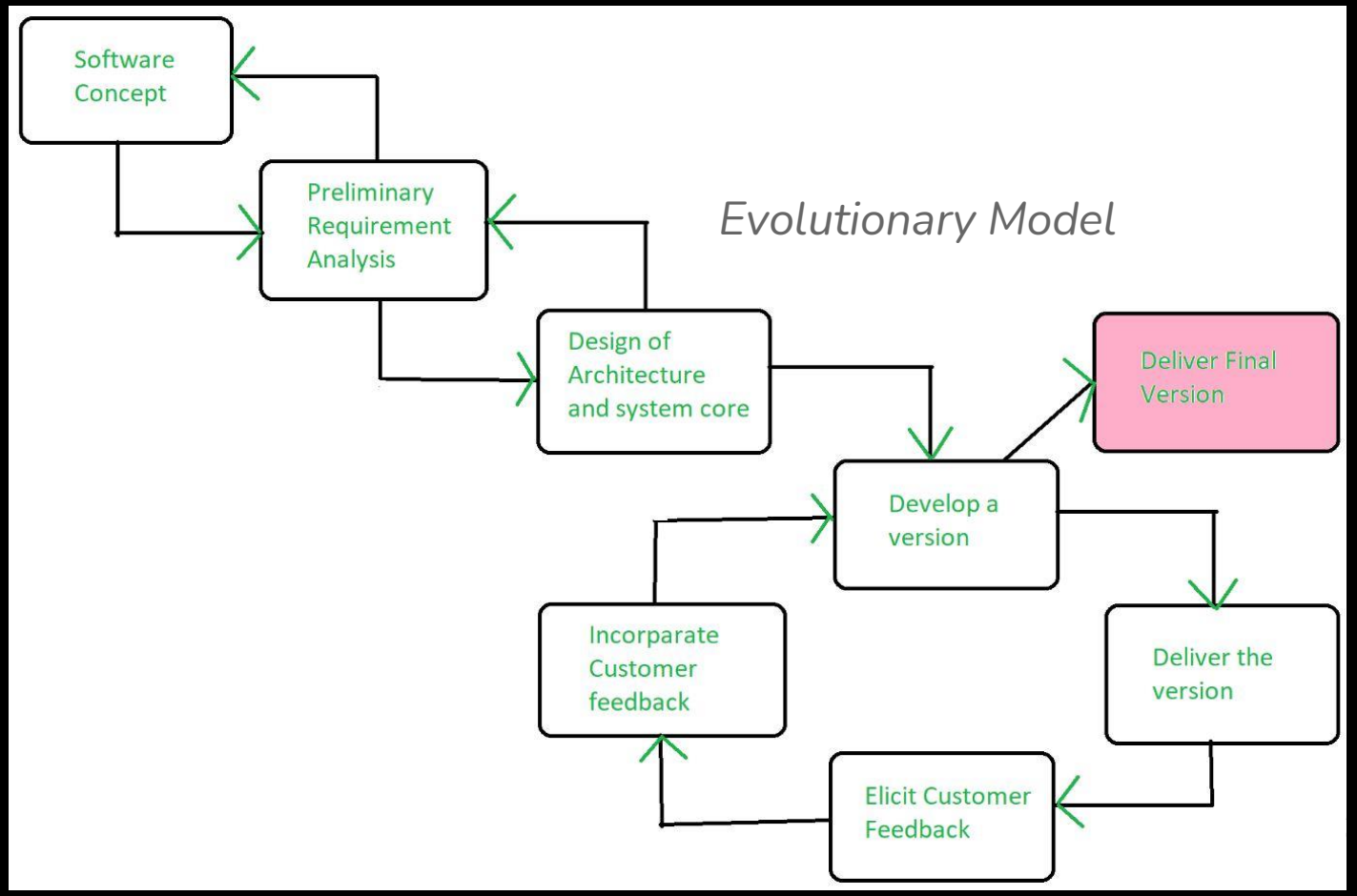
FIVE DRAWBACKS OF RAD MODEL

- For large and scalable projects (continuous development) RAD model requires more human resources to create more number of teams.
- If developers and customers are not committed to the rapid-activities that are necessary to complete the system within a short-time, the entire system will fail.
- If a system cannot be properly modularized, the RAD will be problematic.
- RAD may not be suitable if technical risks are high.
- If performance is an issue, and high performance can be achieved through tuning the interfaces among components, the RAD is useless

EVOLUTIONARY PROCESS MODELS

- The evolutionary model is a combination of the Iterative and Incremental models of the software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done. It is better for software products that have their feature sets redefined during development because of user feedback and other factors. This article focuses on discussing the Evolutionary Model in detail.
- **What is the Evolutionary Model?**
- The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users can get access to the product at the end of each cycle.
- Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan, or process.
- Therefore, the software product evolves with time.
- All the models have the disadvantage that the duration of time from the start of the project to the delivery time of a solution is very high.
- The evolutionary model solves this problem with a different approach.
- The evolutionary model suggests breaking down work into smaller chunks, prioritizing them, and then delivering those chunks to the customer one by one.

- The number of chunks is huge and is the number of deliveries made to the customer.
- The main advantage is that the customer's confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements.
- The model allows for changing requirements as well as all work is broken down into maintainable work chunks.



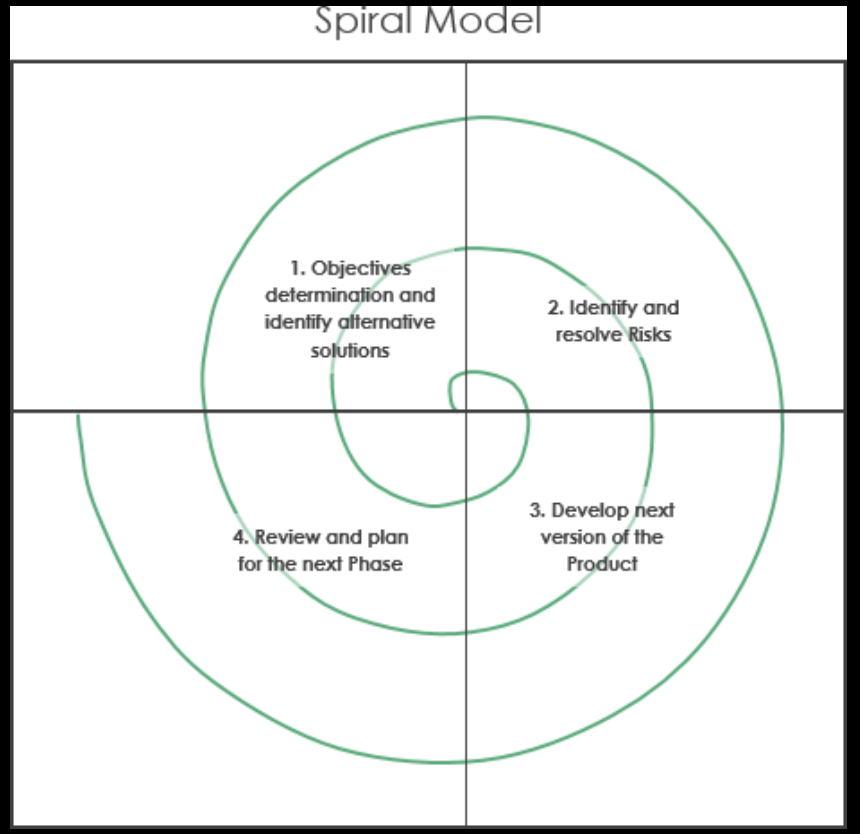
- **Application of Evolutionary Model**
- It is used in large projects where you can easily find modules for incremental implementation. Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.
- Evolutionary model is also used in object oriented software development because the system can be easily portioned into units in terms of objects.
- **Necessary Conditions for Implementing this Model**
- Customer needs are clear and been explained in deep to the developer team.
- There might be small changes required in separate parts but not a major change.
- As it requires time, so there must be some time left for the market constraints.
- Risk is high and continuous targets to achieve and report to customer repeatedly.
- It is used when working on a technology is new and requires time to learn.

ADVANTAGES EVOLUTIONARY MODEL

- **Adaptability to Changing Requirements:** Evolutionary models work effectively in projects when the requirements are ambiguous or change often. They support adjustments and flexibility along the course of development.
- **Early and Gradual Distribution:** Functional components or prototypes can be delivered early thanks to incremental development. Faster user satisfaction and feedback may result from this.
- **User Commentary and Involvement:** Evolutionary models place a strong emphasis on ongoing user input and participation. This guarantees that the software offered closely matches the needs and expectations of the user.
- **Improved Handling of Difficult Projects:** Big, complex tasks can be effectively managed with the help of evolutionary models. The development process is made simpler by segmenting the project into smaller, easier-to-manage portions.
-

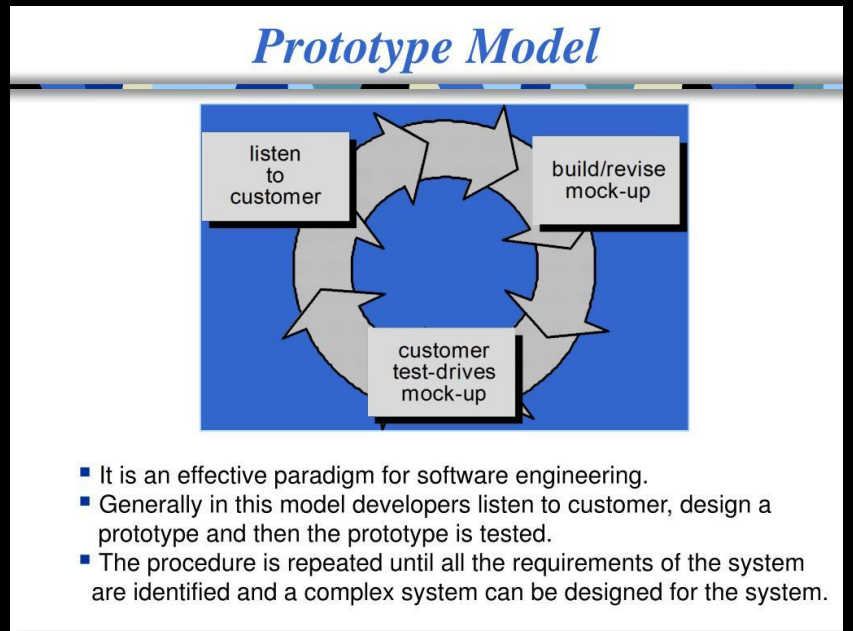
SPIRAL MODEL :

- The spiral model, first described by Barry Boehm in 1986, is a risk-driven software development process model which was introduced for dealing with the shortcomings in the traditional waterfall model. A spiral model looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. This model supports risk handling, and the project is delivered in loops. Each loop of the spiral is called a Phase of the software development process.
- The initial phase of the spiral model in the early stages of Waterfall Life Cycle that is needed to develop a software product. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using a spiral model.



THE PROTOTYPING MODEL

- Consider the following two different situations:
- Often, a customer defines a set of general objectives, but does not identify a detailed specifications like input, processing and output requirements. Its a problem in customer side.
- Next is, problems in developer side; a developer may unsure about the efficiency of an algorithm; the adoptability of an operating system; problems in human-computer interaction etc.
- In these non-stabilized cases, the prototyping model may offer a best solution



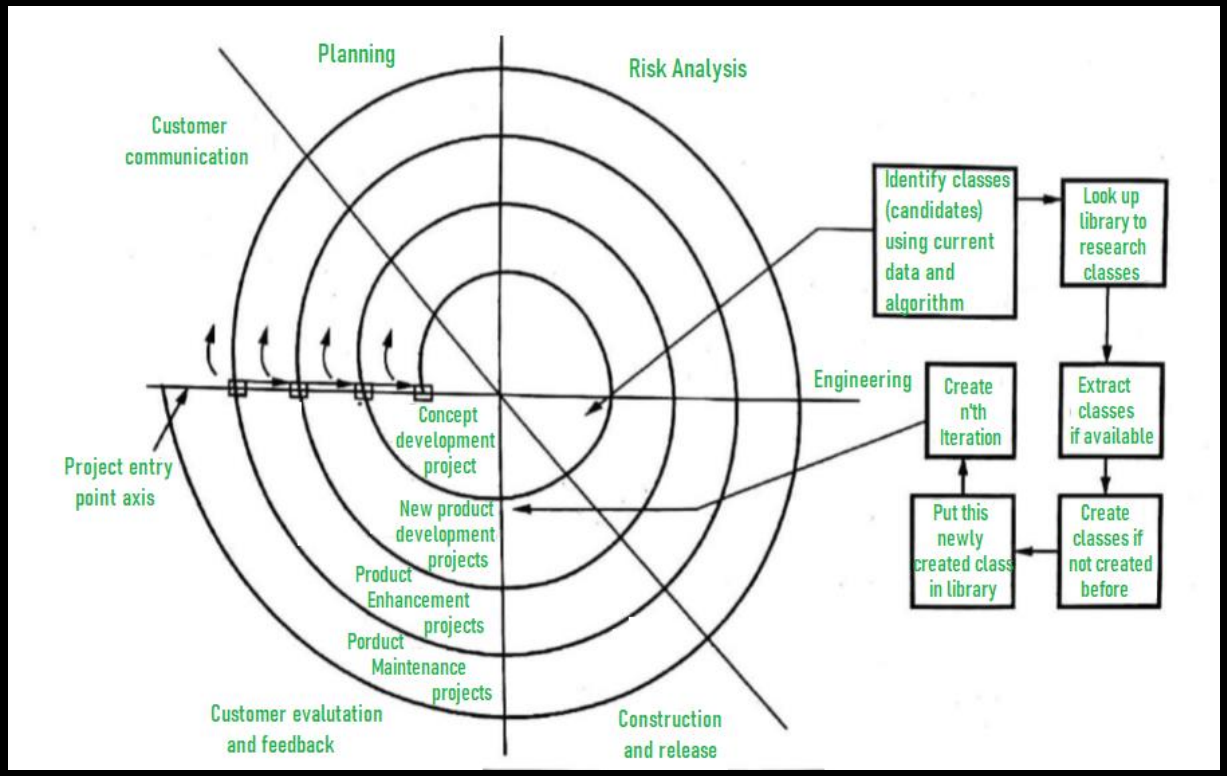
SPECIALIZED PROCESS MODEL

- Special process models take many features from one or more conventional models. However these special models tend to be applied when a narrowly defined software engineering approach is chosen.
- Types in Specialized process models:
- Component based development (Promotes reusable components)
- The formal methods model (Mathematical formal methods are backbone here)
- Aspect oriented software development (Uses crosscutting technology)

COMPONENT BASED DEVELOPMENT

The component based development model incorporates many of the characteristics of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software.

However, the model focuses on prepackaged software components. It promotes software reusability.



- Modeling and construction activities begin with the identification of candidate components. Candidate components can be designed as either conventional software modules or object oriented packages.
- Component based development has the following steps:

- Available component based products are researched and evaluated for the application domain.
- Component integration issues are considered.
- A software architecture is designed to accommodate the components.
- Components are integrated into the architecture.
- Comprehensive testing is conducted to ensure proper functionality.

THE FORMAL METHODS MODEL

The formal methods model encompasses a set of activities that leads to formal mathematical specification of software. Formal methods enable a software engineer to specify, develop and verify a computer system by applying rigorous mathematical notation.

When mathematical methods are used during software development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.

Formal methods provide a basis for software verification and therefore enable softw

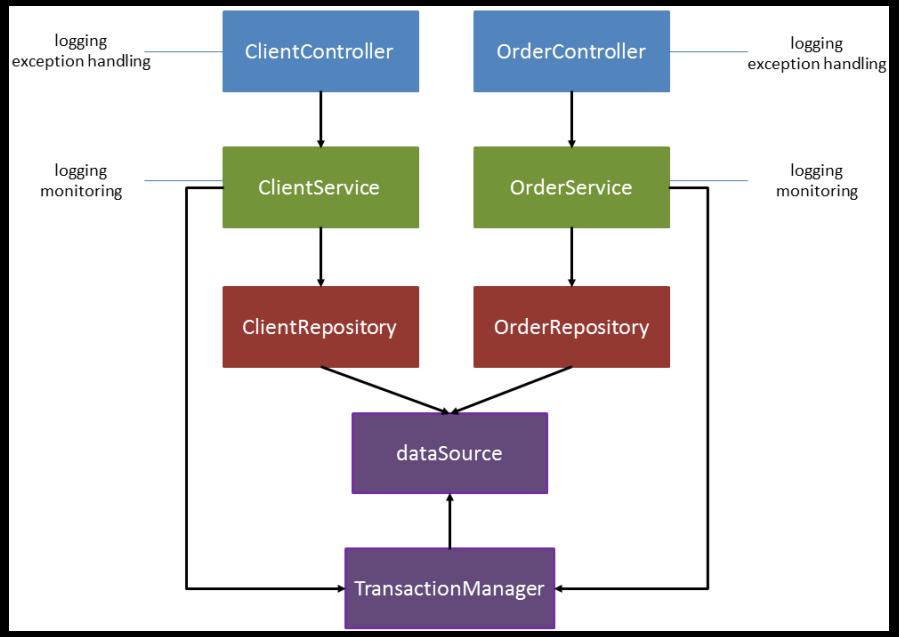
are engineer to discover and correct errors that might otherwise go undetected

PROBLEMS IN FORMAL METHODS MODEL

- The development of formal models is currently time-consuming and expensive.
- Because few developers have the necessary background knowledge to apply formal methods, extensive training is required to others.
- It is difficult to use this model as a communication mechanism for technically unsophisticated people.

ASPECT ORIENTED SOFTWARE DEVELOPMENT

- Aspect Oriented Software Development (AOSD) often referred to as aspectoriented programming (AOP), a relatively new paradigm that provides process and methodology for defining, specifying designing and constructing aspects.
- It addresses limitations inherent in other approaches, including object-oriented programming. AOSD aims to address crosscutting concerns by providing means for systematic identification, separation, representation and composition.
- This results in better support for modularization hence reducing development, maintenance and evolution costs.



THE UNIFIED PROCESS

- The Unified Process (UP) is an attempt to draw on the best features and characteristics of conventional software process models, but characterize them in a way that implements many of the best principles of agile software development.
- The Unified Process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system.
- It helps the architect to focus on the right goals, such as understandability, reliance to future changes, and reuse

AGILE SOFTWARE DEVELOPMENT

- What is agile software development?
- "Agile Development" is an umbrella term for several iterative and incremental software development methodologies. The most popular agile methodologies include Extreme Programming (XP), Scrum, Crystal Clear, Dynamic Systems Development Method (DSDM), and Feature-Driven Development (FDD).

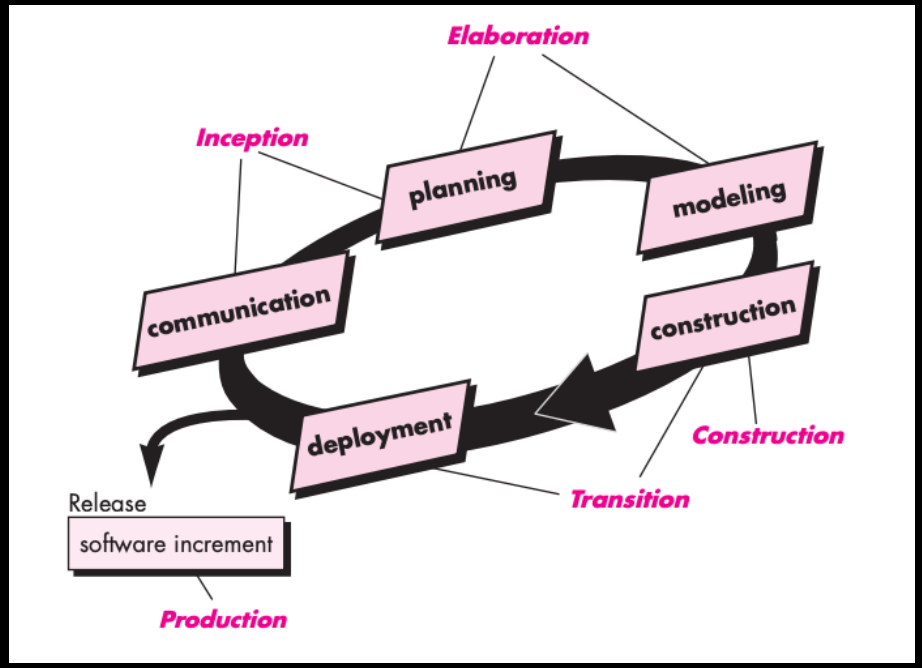
Agile Manifesto:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

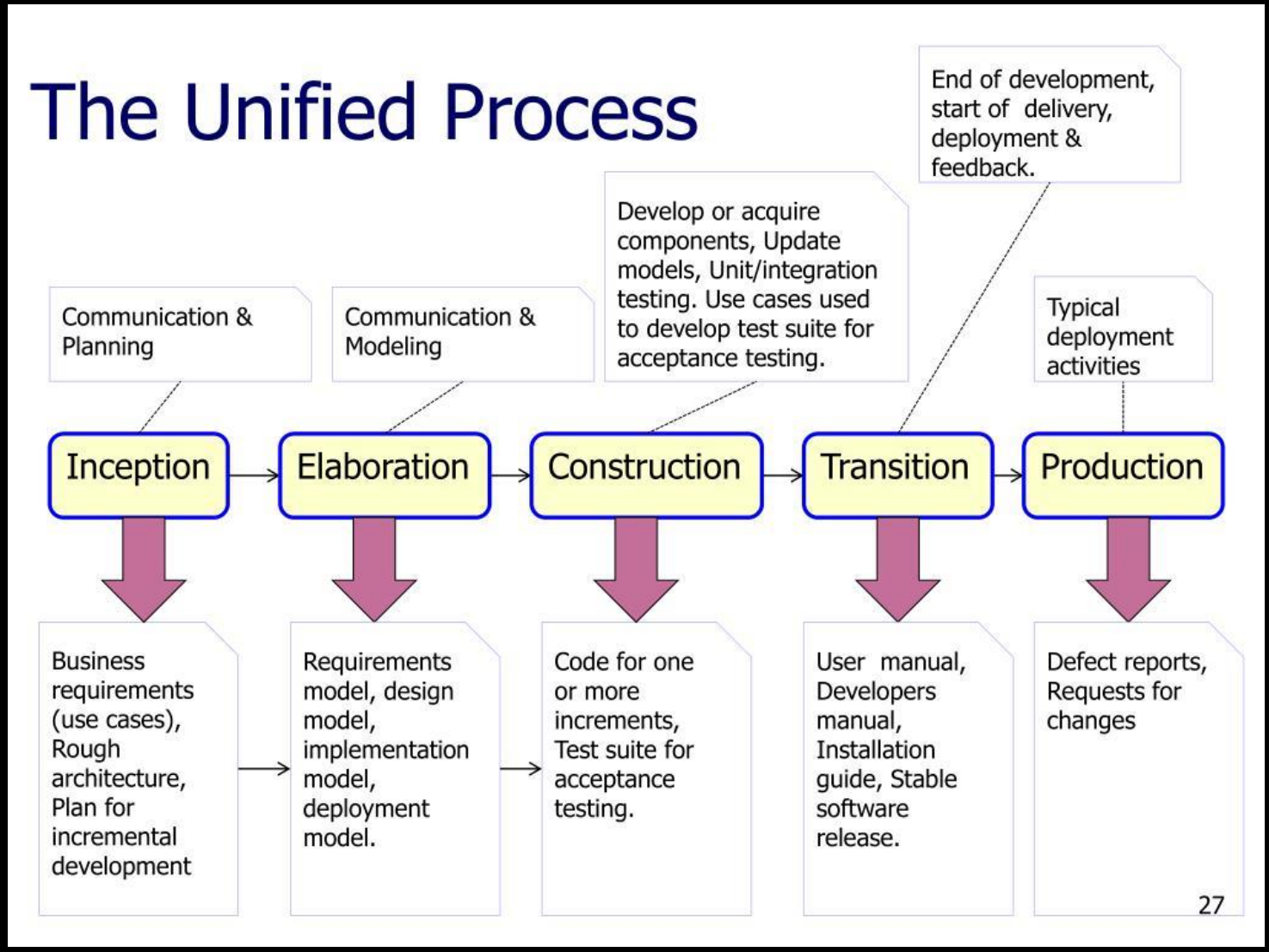
Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation.

PHASES OF THE UNIFIED PROCES

- Inception phase: encompasses both customer communication and planning activities.
- Elaboration phase: encompasses planning and modeling activities. Construction phase: is identical to the construction activity. Transition phase: encompasses latter stages of construction activity and first part of the generic deployment activity.
- Production phase: coincides with the deployment activity of the generic process.

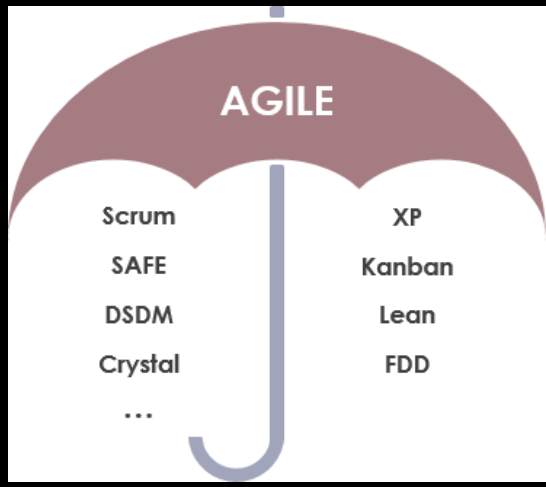


MAJOR WORKFLOW PRODUCED FOR UNIFIED PROCESS

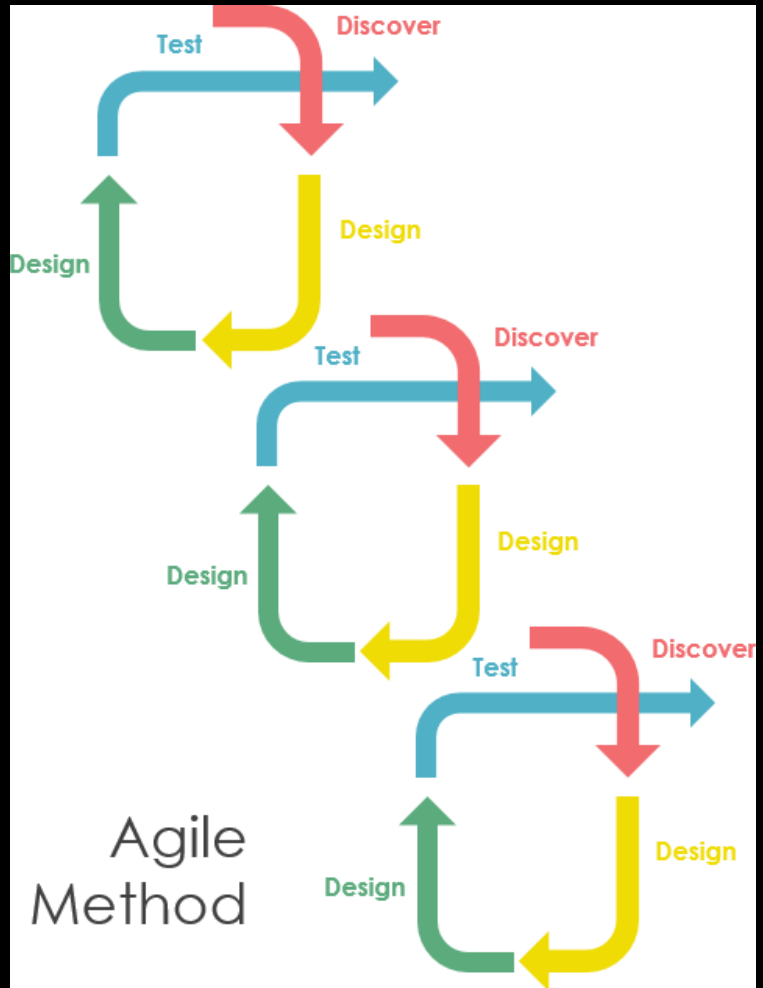


AGILE MODEL

- Agile is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto that is a way of thinking that enables teams and businesses to innovate, quickly respond to changing demand, while mitigating risk. Organizations can be agile using many of the available frameworks available such as Scrum, Kanban, Lean, Extreme Programming (XP) and etc.
- The Agile movement proposes alternatives to traditional project management. Agile approaches are typically used in software development to help businesses respond to unpredictability which refer to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

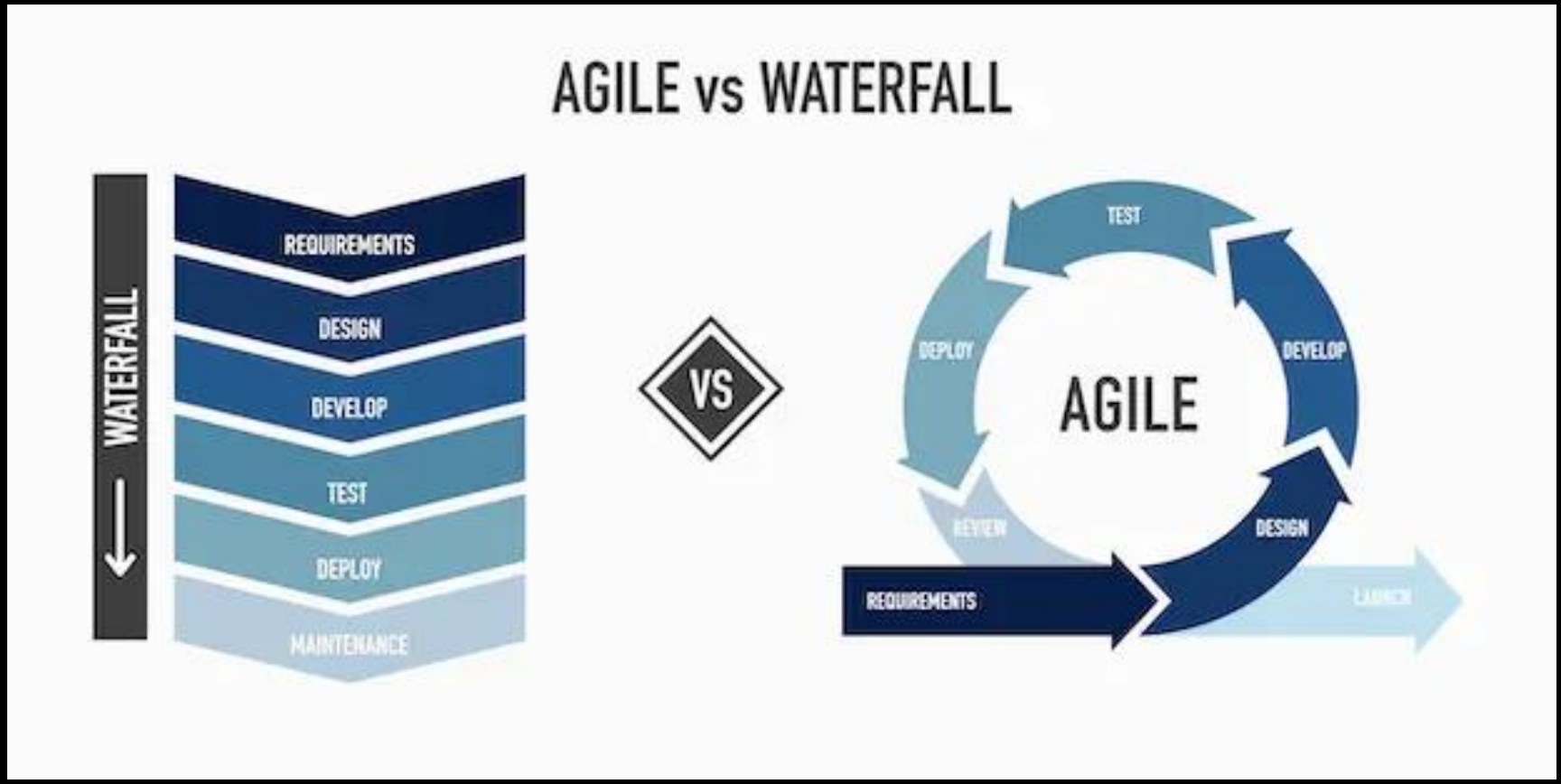


- The primary goal of being Agile is empowered the development team the ability to create and respond to change in order to succeed in an uncertain and turbulent environment. Agile software development approach is typically operated in rapid and small cycles. This results in more frequent incremental releases with each release building on previous functionality. Thorough testing is done to ensure that software quality is maintained.



WHAT ARE THE 12 AGILE MANIFESTO PRINCIPLES?

- *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
- *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
- *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
- *Business people and developers must work together daily throughout the project.*
- *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
- *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
- *Working software is the primary measure of progress.*
- *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
- *Continuous attention to technical excellence and good design enhances agility.*
- *Simplicity — the art of maximizing the amount of work not done — is essential.*
- *The best architectures, requirements, and designs emerge from self-organizing teams.*
- *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*


AGILE vs WATERFALL
WATERFALL
REQUIREMENTS
DESIGN
DEVELOP
TEST
DEPLOY
MAINTENANCE
VS
AGILE
TEST
DEVELOP
DESIGN
DEPLOY
REVIEW
REQUIREMENTS
LAUNCH